Name Matching Across Datasets

-

Text Analytics Model
Proof of Concept

By

Centre of Excellence in Artificial Intelligence

National Informatics Centre

July 2020

Table of Contents

1. Objective of the Exercise:

Personal records across datasets usually is non standardized. Given two names, Objective is to the similarity between them so that information across datasets can be consolidated. With that in mind for farmer records across different schemes in Government of India were subject of similar name checking using a few attributes that might match across datasets like state, district, village etc.. and others more personal identity like date of birth or age, gender, ID No. like Aadhar if available etc..

Farmer Name Datasets & Land Records(LR) Dataset were provided from Gujarat, Maharashtra, Odisha & UP. Land records data was in regional languages and was translated to English using phonemes. It was to be compared with PM Fasal Bima Yojana, PM Kisan and Soil Health Card where the names were in English, with the objective of consolidating Farmer records across datasets.

2. Challenges in Name Matching across datasets :

Name-matching is the difficult task due to following variants:-

(a) Phonetics Similarity
Same name can be written in different forms.
e.g Sourabh, Saurab, Sorav
Avinash, Abhinash
Vikas, Bikash

(b) Missing Space
Name may/may not have space between them
e.g Vinit kumar, Vinitkumar,
Ram Samantaray, Ram Samant Ray

(c)Missing Components
Some times some part of name is not present.
e.g Ravi Singh Chouhan, Ravi Chouhan
Ravi lal Singh, Ravi Singh
P Arun, Arun

(d) Out of order Components
Dataset may have either Surname first or last
e.g. Kumar Swami Iyer, Swami Kumar Iyer, Iyer Swami

(e) **Initials/Full-name**

Name can be written in various form by replacing them with initials
e.g. S B Singh, Shyam Bharti Singh, Shyam B Singh, S Bharti Singh etc

(d) Prefix/Suffix:

Name can have suffix/prefix added, though it may/may not be part of name
e.g Mr, Shri, Ms , ji, Bhai, Ben, Bai, Bhau, Dei, Dada, kumar, kumari etc
Some time they are also part of name e.g **Ji**jabai, Rita**ben,** Ful**kumari** etc..

(e) Maximum Part Matching

Two different name can have more matching than Two simmilar names

e.g. Ram Kumar Bandopadhya, Ravi Kumr Bondopadh:

Here names are of different person but their similarity scores will be high as large fraction of name matches.

These challenges often comes together making name matching more tricky. For modelling , we will need the dataset capturing all these variety.

3. Data Exploratory Analysis :

Initially data of Land Record, PMFBY, PMKISAN of few villages of UP, Maharashtra & Odisha each was given. Later Land Record (LR), PMFBY, PMKISAN and Soil Record of Gujarat was provided. Datasets were analysed for finding missing values, unique values, common attributes etc. Some examples are given in Figure 1 & Figure 2.

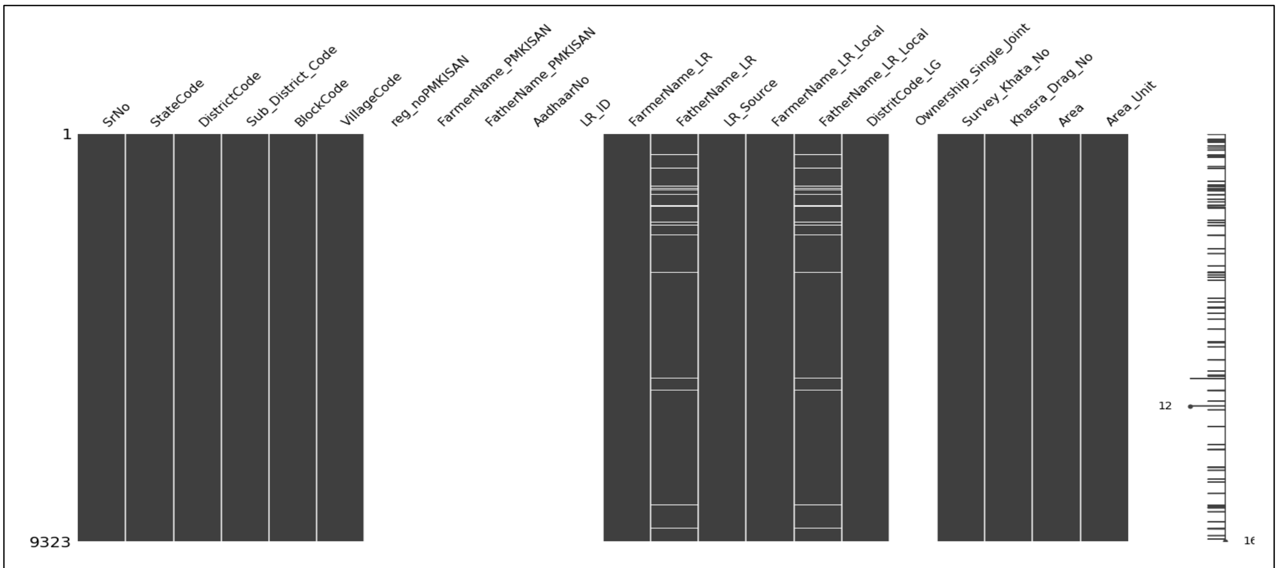

Figure 1 - Some common/ similar Attributes in Odisha PM_KISAN & Land Record

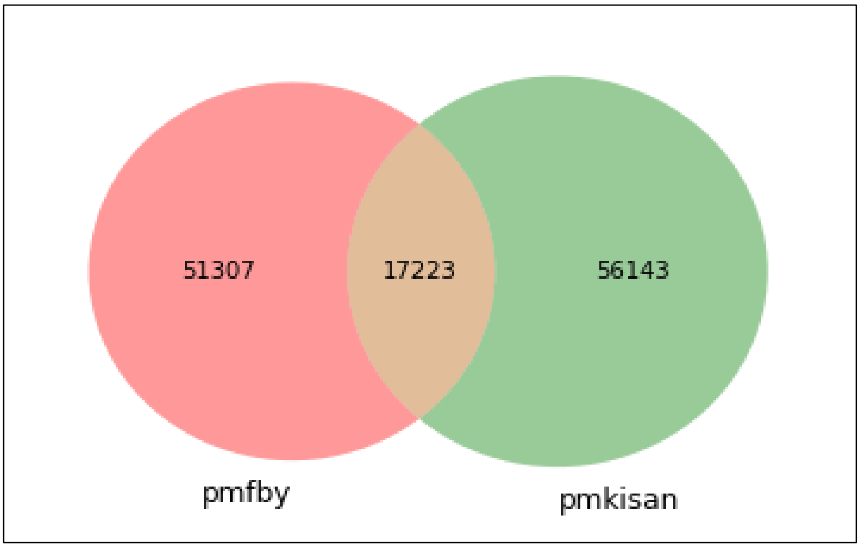

Figure 2 - Venn Diagram: Adhaar No. distribution of PMFBY & PMKISAN of Odisha

On analysis it was found that there is no matching of Survey number, land division number between 2 dataset(LR & PMFBY).

We found that matching can be done on the basis of Name, location (village-code / block-code / district-code / state-code) & gender only as other field are either data missing or not available.

For Matching Names we needed positive and negative samples.

Positive Samples:
Samples obtained from PMFBY (PM Fasal Bima Yojana) & PMKISAN based on same Aadhar Number were extracted. From these samples, manual checking of similarity and labelling was done. This step generated mostly Positive samples and a few wrong ones.

Negative Samples:
Other then Aadhar based matching, for matching with records which did not have aadhar, other attributes such as location (codes) & gender which can be compared easily were used, and then we only required method to compare names. Fuzzy Name Matching using Machine learning was used. Farmer Names From PMFBY & PMKISAN were extracted and compared with each other on the basis of fuzzy phonetic similarity (Soundex).

Preparing Negative Samples is a difficult task as the sample must be representative of its entire distribution for effective ML modelling.

Negative Samples was generated in following steps:

1.Name-pairs were generated by pairing every unique names in our dataset. This generated a lots of pairs.

2. To facilitate manual labelling and to have dataset of wide-variety, the fuzzy-soundex-similarity score was calculated on these pairs.

This made manual labelling name-pair easier as :

(a) pairs having low fuzzy-soundex-similarity (<60) can be labelled as 'not same' by mere looking .

(b) pairs having fuzzy-soundex-similarity (>60 and <95) required more attention in labelling.

(c) pairs having fuzzy-soundex-similarity (>95) are mostly equal and can be labelled 'same' easily.

Following is the final generated dataset distribution

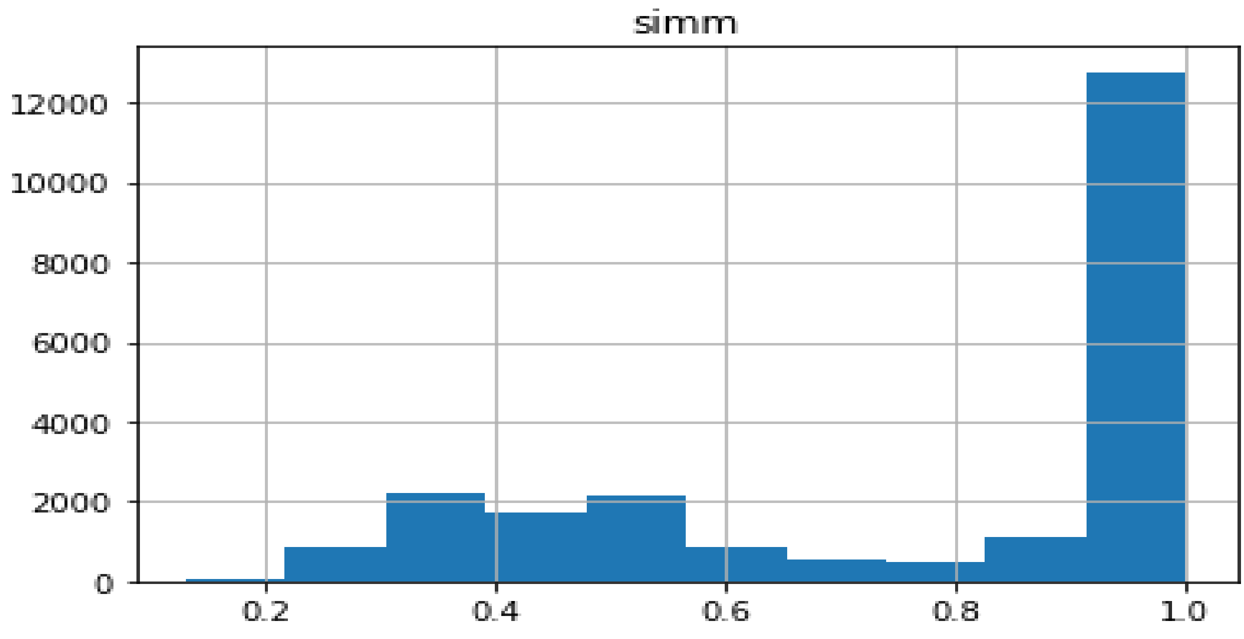

Fig.3 - graph denoting distribution of samples on the basis of simm: fuzzy phonetic similarity (Soundex). X Axis : fuzzy-soundex-similarity & Y Axis : #name-pairs

On Initial datasets, sample data check carried out shows a left tailed distribution. Since most of the sample name-pairs were clustered with similarity score between 0.85-1, we get large number of positive name-pairs having high fuzzy-soundex-similarity. So the distribution showed that further work can be carried out.

4. Data Pre-Processing :

Following steps were performed for cleaning the name attributes in the datasets.

i. Attributes Extraction –

Extract Name with attributes like village code, gender, Father Name from Database-1. Do same for Database-2

ii. Data Cleaning –

It includes:

(a) Making Names lower case. Removing unnecessary characters like .(dot),/,- etc.

(b) Name of Certain Region contains suffix e.g.

In Maharashtra: Bhau, Rao

In Gujarat: Bhai, Ben

In North India: Kumar, ji

these can be removed as these suffix increase matching scores. Common Suffixes can be found using analytic or manually input

(c) Standardizing Village code, Gender etc..

(A) Name cleaning

- remove numeric words and special characters
- lowercase all character

(B) Stop-word Removal

- Salutation removal e.g smt, shri, mr, dr, ms etc
- common word removal e.g. 'bhai', 'bhau', 'bhoi', 'bai', 'kumar', 'kumr', 'kmr', 'ben', 'dei', 'devi', 'debi',kumaar'
- common suffix removal from word. e.g 'saheb', 'kumar', 'kumaar', 'bhai', 'bhau', 'bai', 'ben', 'bai','sab'

(C) Name Standardization

Names are standardized according to Indian context.

1. Replace e by I (इ, ई):

e.g. eshwar → ishwar,

2. Replace adjacent similar character by single character

e.g. raaghaav → raghav

3. replace Unigrams:
v → w
j → z
q → k

e.g.
raghav → raghaw
vinod → winod
rav → raw
jakir → zakir
quran → kuran

4. replace bigrams :
ph → f
th → t
dh → d
sh → s
ck → k
gh → g
kh → k
ch → c
e.g.
phogat → fogat
yatharth → yatart
parth → part
dhoni → doni
harish → haris
wickas → wikas
raghaw → ragaw
khaton → katon
choubey → coubey

5. Replace(ह):
ah → h
e.g. allah → alh
maharana → mharana

6. Remove a if previous char is not i,o,u (consonant + a = consonant)
e.g. mharana → mhrn

(D) Common part removal

Name pairs is split on space and common word is removed.

e.g Ram Manohar Singh → rm mnohar sing →rm

Syam Manohar Singh → sym mnohar sing → sym

iii. Encoding Generation -

Generating small length encoding of names capturing phonetic property. Used Modified Soundex for Encoding Generation.

(a) Modified Soundex for Indian context-

Soundex is modified for improving the matching.

encoding: alphabets

0: 'aeiouvyhw',

1: 'kgqc',

2: 'cj',

3: 'td',

4: 'jzx',

5: 'm',

6: 'pfbwv',

7: 'l',

8: 's',

9: 'r',

'!': 'n',

e.g. **ramesh chandra swain → rms cndr swn{{'958'},{'1!39', '2!39'}, {'8!', '86!' }}**

(b) common soundex encoding is removed.

e.g. Name1: {{'958'},{'19', '2!39'}, {'8!', '86!' }} → {{'958'}, {'8!', '86!' }}

Name2: {{'58'},{'2!39', '3!39'}, {'5!', '6!' }} → {{'58'}, {'5!', '6!' }}

'2!39' is common in both name, so common encoding set is removed as shown.

iv. Machine Learning Pipeline:

Different algorithms were explored for calculating the names similarity distance score. Entire Process is divided into 2 passes:

Pass – 1:
This Pass generates candidate pairs from the input Database. As there lacs of records every record in each dataset, all combinations cannot be checked directly.

Pass – 2:
After getting candidate pairs from Pass-1, applies ML model for classification

Steps followed :

a. **Candidate Pair Generation -**

If we directly do cross product of names in 2 Name-list we will get a huge no. of candidates. e.g. If 2 Databases are of size 50K, we will get 250 crore name pairs, which will make features generation and name matching process time consuming.

So we filter the name on the basis of Village code, Gender etc. across datasets. It is less computationally intensive matching algorithm with low threshold applied to further reduce candidate pairs.

e.g. 2 names are compared only if they belong to same village.

Fuzzy soundex with **threshold of 50** was used to get candidate pairs. JaroWinkler can be used as it has less computation complexity.

b. **Features Generation** –

Similarity Scores of different algorithm such as Jaro-Winkler, Jaccard, Cosine similarity etc was generated. Various string similarity measures are analyzed both on raw names as well as processed names. Some of these measures analysed were :-
Edit based:

- Hamming
- MLIPNS
- Levenshtein
- Damerau-Levenshtein
- Jaro-Winkler

Token based :

- Jaccard index
- Overlap coefficient

Sequence based:

- longest common subsequence similarity
- longest common substring similarity
- Ratcliff-Obershelp similarity

Simple:

- Prefix similarity

Postfix similarity

- Length distance
- Identity similarity
- Matrix similarity

Phonetic:

- Soundex Similarity

	Farmer_Name_x	Farmer_Name_y	label	Jaro-Winkler	Damerau-Levenshtein	MLIPNS	Hamming	Overlap	Jaccard
147	biranchi kumar behera	biranchi behera	1	0.916190	0.714286	0.0	0.476190	1.000000	0.71428
18868	manoj kumar rout	chakradhara pradhan	0	0.508041	0.157895	0.0	0.052632	0.437500	0.25000
2263	brajabandhu sundaray	brajabandhu sundaray	1	1.000000	1.000000	1.0	1.000000	1.000000	1.00000
20888	dushasan patra	satrughan pradhan	0	0.720015	0.352941	0.0	0.000000	0.928571	0.72222

LCSSeq	LCSStr	Ratcliff-Obershelp	Soundex_prune	Soundex_simple	Prefix	Postfix	Length	fuzzywuzzy
0.714286	0.428571	0.833333	0.80	0.80	0.428571	0.333333	0.714286	1.0
0.315789	0.105263	0.114286	0.35	0.59	0.000000	0.000000	0.842105	0.3
1.000000	1.000000	1.000000	1.00	1.00	1.000000	1.000000	1.000000	1.0
0.470588	0.235294	0.516129	0.40	0.53	0.000000	0.000000	0.823529	0.5

Figure 4 – A few samples showing different String Similarity Measures

c. **Training the Model –**

The trained model will predict the pairs are similar or not based on above features.

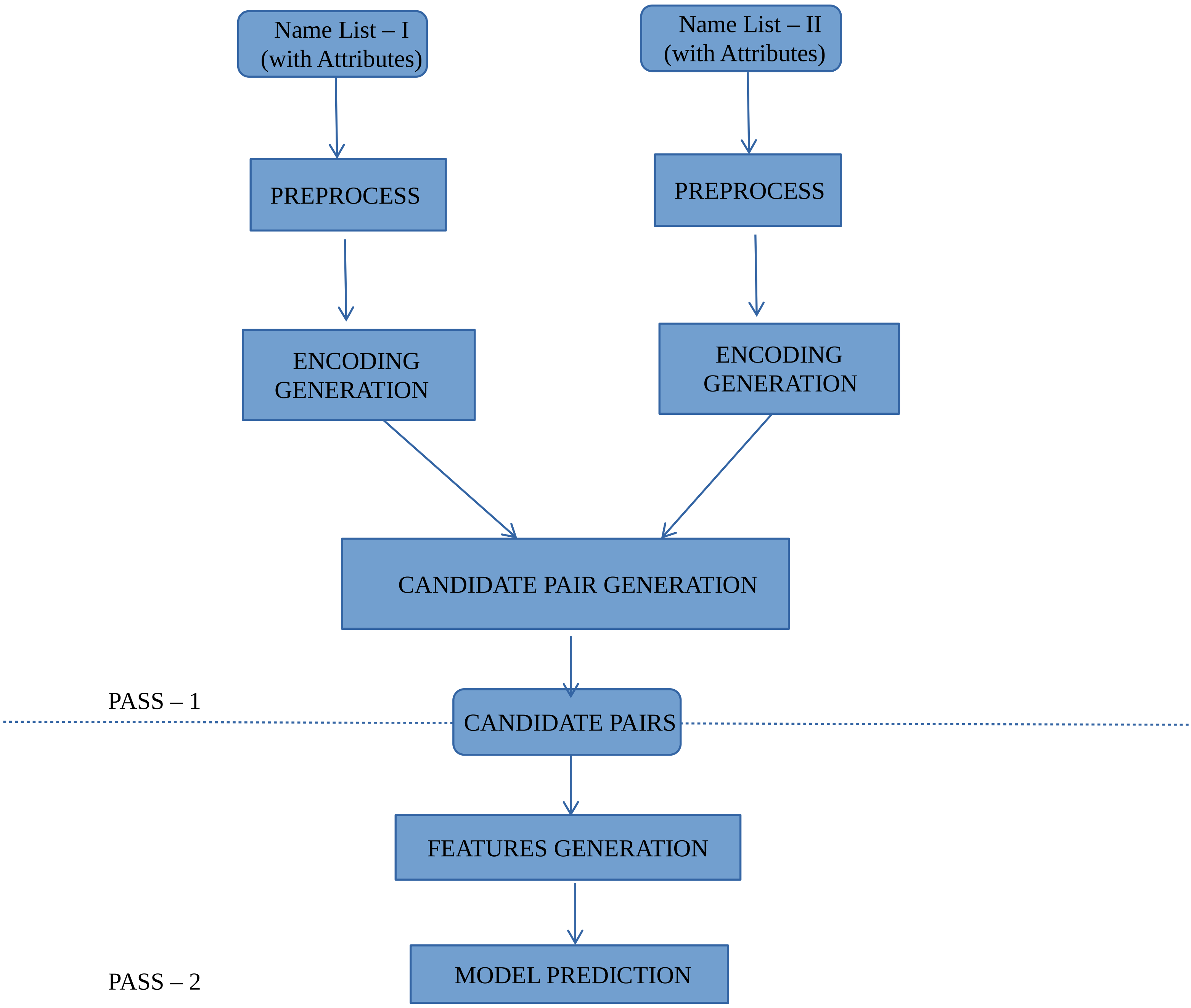

Figure 5 : Dataflow Pipeline for Name Similarity Matching

v. Machine Learning:

To improve model accuracy, XGBoost, a Gradient boosting algorithm was used on these similarity metrices scores.

XGBoost

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way.

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

a. Initial Model -

In Initial model pipeline, in phase I only name cleaning was done. These names were fed to similarity measure as shown to generate features. These features are input to XGBOOST Algorithm.

Training Parameters (Default):

base_score=0.5, booster='gbtree', colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1, gamma=0, learning_rate=0.1, max_delta_step=0, max_depth=3, min_child_weight=1, missing=None, n_estimators=100, n_jobs=1, nthread=None, objective='binary:logistic', random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None, subsample=1

Data set was randomly divided into **60% ,40% for trainset** & **testset** respectively

Metrics used for the XGBoost Algorithm – f(i)

'Jaro-Winkler',
'Damerau-Levenshtein',
'MLIPNS',
'Hamming',
'Overlap',
'Jaccard',

'LCSSeq',
'LCSStr',
'Ratcliff-Obershelp',
'Soundex_prune',
'Soundex_simple',
'Prefix',
'Postfix',
'Length',
'fuzzywuzzy'

Result on UP, Maharashtra & Odisha dataset (Small dataset) by Initial Model.

Accuracy: 99.68%

precision_score : 0.9971783295711061

recall_score : 0.9979667909183327

confusion_matrix:

[[4732   25]

 [18 8835]]

f1_score: 0.99757240444871

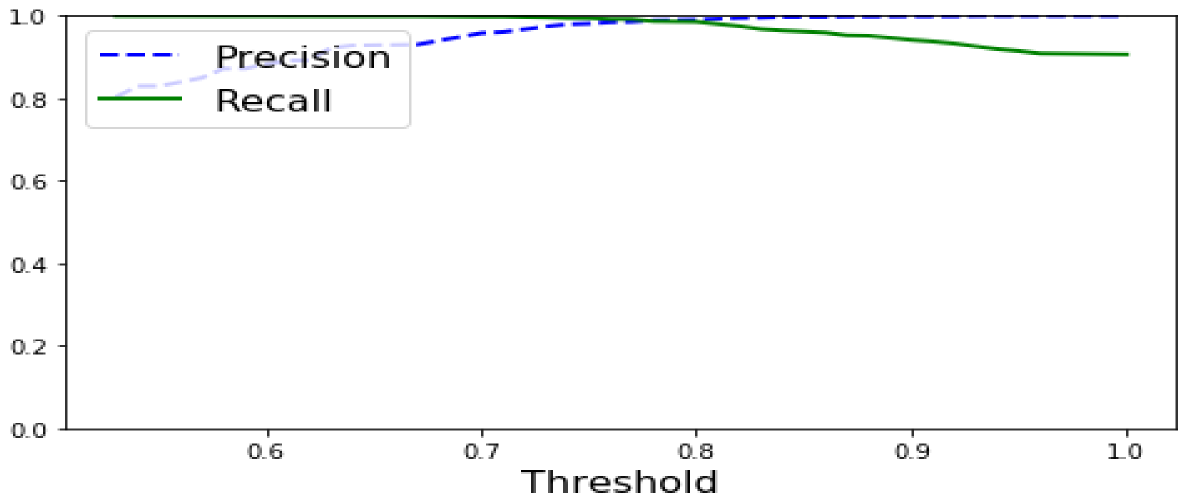

Figure 6: precision & recall vs threshold

High Precision & Recall on a small variuant of regional dataset doesn't mean that it will extrapolate well to All India Data having different regional nuances. However, High precision & recall for a district within a state will scale well to the entire state.

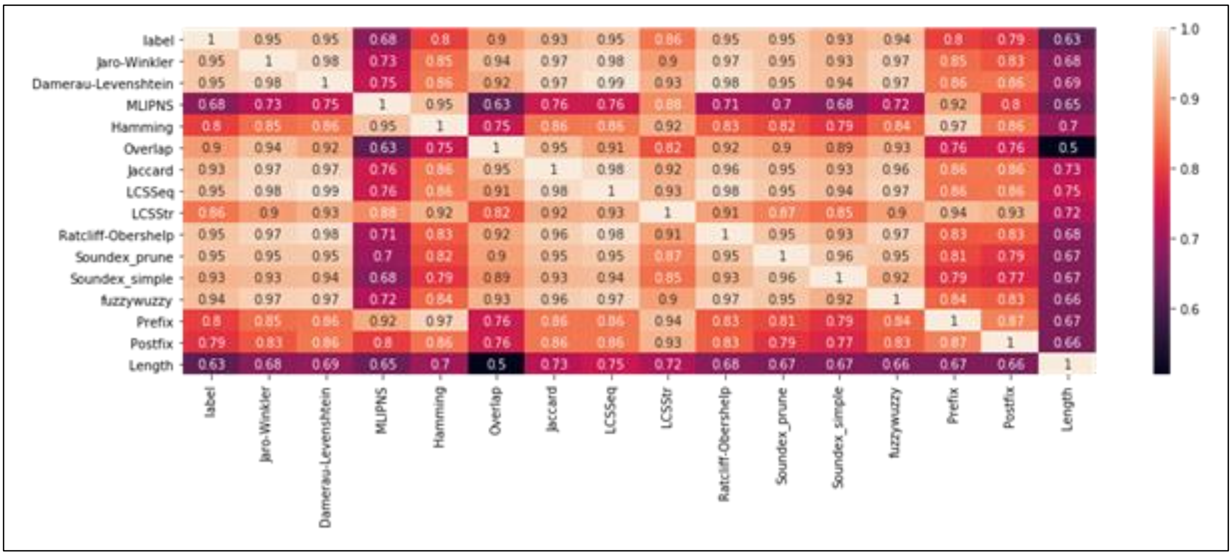

Figure 7: Correlation between the different metrics

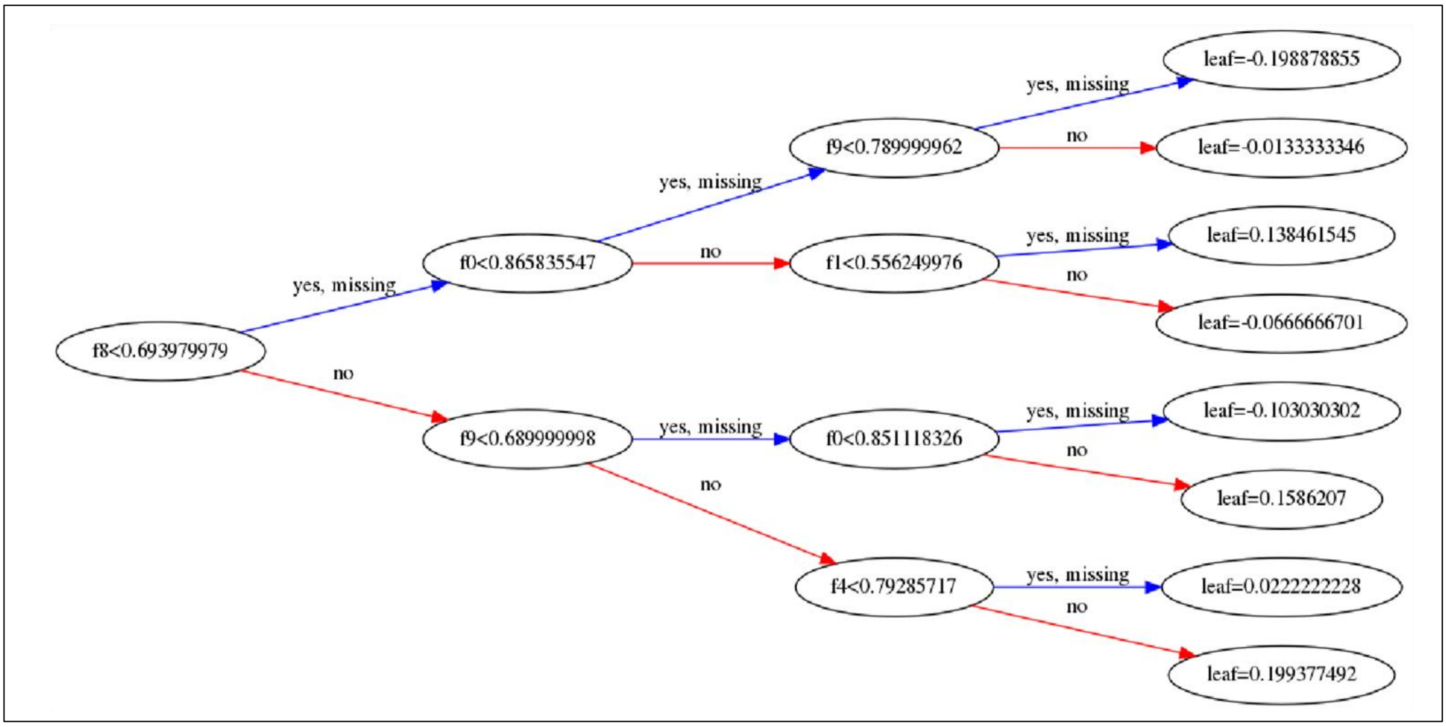

Figure 8: one of the Decision Tree in XGBoost, fi denote the ith metrics as above

Limitation:

However model was not able to perform well on huge Gujarat Dataset as model had not considered all variants of namepair that may exist in regional datasets. To solve this whole pipeline was redesigned to account the challenges in name matching.

Modified Soundex algorithm result -

Alternately Modified Soundex algorithm was also tried & not taking any other similarity measures.

Similarity Threshold:0.80

Accuracy: 98.41%

precision_score : 0.990236148955495

recall_score : 0.9852027561278662

confusion_matrix: [[4671 86]
[131 8722]]

f1_score: 0.9877130400317083

Result:

- XGBoost model performed slightly better than Modified Soundex algorithm. Minute increase (~1%) in XGBoost model accuracy & F1 score with increase in complexity. However this is **dependent on dataset available.**

Limitation:

- Dataset is skewed. Better the data better will be model
- Other string similarity metrics can also be added to increase further accuracy
- Much slower than Modified Soundex algorithm. Some metrics can be eliminated\ dimension reduction techniques can be used to speed up the processing

b) Final Model -

Following were the features generated by using selected Similarity Measures in the final model.

'SOUNDEX_SIMM':

all combination of soundex encoded name pair are generated and compared using Radclif-Obershelp similarity

'SOUNDEX_PARTIAL_SIMM':

all combination of soundex encoded name pair are generated and shorter name is compared with clipped longer name of same length using Radclif-Obershelp similarity.

'PARTIAL_MATCH_NAME':

all combination of standardized name pair are generated and shorter name is compared with clipped longer name of same length using Radclif-Obershelp similarity

'JARO_WINKLER_ONNAME':

Jaro-winkler similarity is applied on all permutation of standardized name pair and maximum value is written

'UNCOMMON_SNDX_LN':

length uncommon soundex of shorter name

'DLVNSTEIN':

Damerau–Levenshtein similarity on standardized name pair is calculated

UNCOMMON_SNDX_LN_RATIO

ratio of uncommon shorter soundexed string and uncommon longer soundex string

SUBSEQUENCE SIMILARITY:

Longest common subsequence is computed on standardized name to calculate sub sequence similarity.

Parameter Tuning:

Parameter Tuning was done by doing grid search on following values:

```
params = {
    'min_child_weight': [1, 5, 10],
    'gamma': [0.5, 1, 1.5, 2, 5],
    'subsample': [0.6, 0.8, 1.0],
    'colsample_bytree': [0.6, 0.8, 1.0],
    'max_depth': [3, 4, 5]
    }
```

Best Model found parameters:

XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
colsample_bynode=1, colsample_bytree=1.0, gamma=2,
learning_rate=0.1, max_delta_step=0, max_depth=4,
min_child_weight=10, missing=None, n_estimators=100, n_jobs=1,
nthread=None, objective='binary:logistic', random_state=0,
reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
silent=None, subsample=0.6, verbosity=1)

Result :

NAME1 Original	NAME2 Original	NAME1	NAME2	NAME1 LIST	NAME2 LIST	UNCOMMON NAME1	UNCOMMON NAME2
Rasanand Pal	rasananda bhoi	rsnnd pl	rsnnd	['rsnnd', 'pl']	['rsnnd']	{'pl'}	set()
Niranjan Mallick	niranjan mallik	nirnzn mlik	nirnzn mlik	['nirnzn', 'mlik']	['nirnzn', 'mlik']	set()	set()
Lakshman Bhoi	lakshman kumar parida	lksmn	lksmn prid	['lksmn']	['lksmn', 'prid']	set()	{'prid'}

snd1	snd2	SNDX U1	SNDX U2	SOUNDEX SIMILARITY	SOUNDEX PARTIAL SIMILARITY	PARTIAL MATCH NAME	JARO WINKLER ONNAME	UNCOMMON SNDX LN	UNCOMMON SNDX LN RATIO	DLVNSTEIN	SUBSEQ NAME
{('67',)}	set()	{('67',)}	set()	1	1	1	1	0	0	3	0.714285714286
set()	set()	set()	set()	1	1	1	1	0	1	0	1
set()	{('693',)}	set()	{('693',)}	1	1	1	1	0	0	5	0.555555555556

Figure 9 : Sample showing different similarity metrices used in XGBoost

Some samples of same name-pairs predicted by model from 2 dataset

Manglu	mangaloo
Patel Maheshbhai	Mahesh Kantilal Patel
Patel Maheshbhai	patel maheshkumar b
Patel Maheshbhai	Patel Mahesh Virji Meghani
Patel Maheshbhai	patel maheshkumar a
Ram Narayan	raam naraayan singh
Munni Lal	munn laal
patel chimanbhai	Patel Chimanlal Ramji
Rajendra Prasad Pal	raajendra prasaad paal
patel chimanbhai	patel chimanlal a
Ram Khelavan	raam khelaavan
Patel Pankaj Kumar	patel pankajkumar d
Sandip Singh	sandeep kumaar singh
Patel Ranchodbhai	patel ranchodabhai m
Hira Lal	heeraalaal
Patel Ranchodbhai	patel ranchodbhai b
Manna Singh	munna singh
Patel Ranchodbhai	Patel Ranchhodbhai Madhavalal
Shivrani	shivaraanee
PATEL MADHUBHAI	patel madhubhaqi m
Vishwanath	vishvanaath
PATEL MADHUBHAI	patel madhubhai m
Patel Parsottambhai Haribhai	patel parasotambhai h
arvindbhai mohanbhai narola	Arvindbhai Mohanbhai
SHANKARBHAI DEVABHAI CHAUDHARY	chaudhari shankarabhai d
Meenakshi Sambhaji Machale	Minakshi Sanbhaji Machle
Vijay Dada Lonkar	Vijay Dada Lonakar
Pravin Baban Kalaskar	Prvin Baban Kalasakar
Bapurav Daulatrav Mokashi	Bapu Daulatarav Mokashi
Sampat Babasaheb Dalvi	Sampat Baba Dalavi
Sampatrao Babasaheb Jagdale	Sampat Baba Dalavi
Akshay Kailas Gandhale	Akshay Kailas Gandhale
Sambhaji Pandurang Arawade	Sanbhaji Pandurang Aravade
Lalita Ashokrao Mokashi	Lalita Ashokarav Mokashi
Sunita Dilip Gandhale	Sunita Dileep Gandhale
Digvijay Vitthalarav Mokashi	Digvijay Vithhal Mokashi
Bhagwan Daya Kale	Bhagawan Daya Kale
Satish Gangadhar Ghadge	Satheesh Gangadhar Ghadge
Yuvraj Jalindar Gandhale	Yuvaraj Jalindar Gandhale

Figure 10: Sample results of farmers' names match across datasets

7. Future Work –

1. User interface can be made, through which:

(a) Degree of recall and precision can be controlled.

(b) Challenges/variants in name can be relaxed or increased

e.g. we can remove or add setting for prediction of out of order names such as bhola ravi, ravi bhola

(c) Some exception-rules / stop-words / salutation etc. can be added or removed.

e.g. in Maharashtra people frequently use Bhau. Such rule can be added to make predicitons more accurate as per the regions.

2. Better similarity features can be explored and implemented.

3. Deep learning based techniques (Siamese network, LSTM etc) can be used – work has been started on this aspect also to check out performance improvement by letting the system do the feature engineering by itself using millions of records available in the datasets, to overcome the limitation having to finetune the model parameters manually according to regional datasets.

This will form the POC of Name Similarity Search Deep learning Exercise in future.